

-  Secrets
-  ABAP
-  Apex
-  C
-  C++
-  CloudFormation
-  COBOL
-  C#
-  CSS
-  **Flex**
-  Go
-  HTML
-  Java
-  JavaScript
-  Kotlin
-  Kubernetes
-  Objective C
-  PHP
-  PL/I
-  PL/SQL
-  Python
-  RPG
-  Ruby
-  Scala
-  Swift
-  Terraform
-  Text
-  TypeScript
-  T-SQL
-  VB.NET
-  VB6
-  XML















Flex static code analysis

Unique rules to find Bugs, Security Hotspots, and Code Smells in your FLEX code

- All rules 76
-  Vulnerability 5
-  Bug 9
-  Security Hotspot 1
-  Code Smell 61

Tags ▾

Search by name... 

Security.allowDomain(...) should only be used in a tightly focused manner		Vulnerability
The flash.system.Security.exactSettings property should never be set to false		Vulnerability
Dynamic classes should not be used		Code Smell
"LocalConnection" should be configured to narrowly specify the domains with which local connections to other Flex application are allowed		Vulnerability
"default" clauses should be first or last		Code Smell
Event types should be defined in metadata tags		Code Smell
Event names should not be hardcoded in event listeners		Code Smell
The special "star" type should not be used		Code Smell
Variables of the "Object" type should not be used		Code Smell
Methods should not be empty		Code Smell
Constant names should comply with a naming convention		Code Smell
All branches in a conditional structure should not have exactly the same implementation		Bug
Classes that extend "Event" should		

Event types should be defined in metadata tags

Analyze your code

-  Code Smell
-  Critical
- 
-  design

According to the Flex documentation :

In an ActionScript file, when you define component events or other aspects of a component that affect more than a single property, you add the metadata tag outside the class definition so that the metadata is bound to the entire class, as the following example shows:

```
// Add the [Event] metadata tag outside of the class
[Event(name="enableChange", type="flash.events.Event")]
public class ModalText extends TextArea {

    ...

    // Define class properties/methods
    private var _enableTA:Boolean;

    // Add the [Inspectable] metadata tag before the
    [Inspectable(defaultValue="false")]
    public function set enableTA(val:Boolean):void {
        _enableTA = val;
        this.enabled = val;

        // Define event object, initialize it, then create it
        var eventObj:Event = new Event("enableChange");
        dispatchEvent(eventObj);
    }
}
```

In this example, the "enableChange" event must be considered part of the API. Therefore, it should be strongly typed.

Noncompliant Code Example

```
[Event(name="enableChange")]
public class ModalText extends TextArea {...}
```

Compliant Solution

```
[Event(name="enableChange", type="flash.events.Event")]
public class ModalText extends TextArea {...}
```

Available In:

sonarcloud  | sonarqube 

<div>override "Event.clone()"</div> <div> Bug</div>
<div>Constructors should not dispatch events</div> <div> Bug</div>
<div>"ManagedEvents" tags should have companion "Event" tags</div> <div> Bug</div>
<div>Objects should not be instantiated inside a loop</div> <div> Code Smell</div>
<div>Two branches in a conditional structure should not have exactly the same implementation</div> <div> Code Smell</div>
<div>Constructor bodies should be as lightweight as possible</div> <div> Code Smell</div>
<div>Only "while", "do" and "for" statements should be labelled</div> <div> Code Smell</div>
<div>Statements, operators and keywords specific to ActionScript 2 should not be used</div> <div> Code Smell</div>
<div>"for" loop stop conditions should be invariant</div> <div> Code Smell</div>
<div>Unused function parameters should be removed</div> <div> Code Smell</div>